Exploring sample attributes

Transformations and model summaries

R.W. Oldford

UNIVERSITY OF WATERLOO

Example: Cosmetic company on Facebook

By 2014, Facebook was the most used social network averaging about 1.28 billion monthly active users.

A cosmetics company had been using a Facebook page they created to reach their customers as well as potential customers.

Likely, the page had been active for a few years, when they decided to investigate the effectiveness of the various postings they had made on their page.

To get some idea, they decided to collect all of the posts they published in 2014 (January 1 to December 31) together with different features recorded on each post.

Questions:

- What is a unit here?
- What is the target population?
- The study population?
- The sample?

Source:

S. Moro, P. Rita and B. Vala (2016). "Predicting social media performance metrics and evaluation of the impact on brand building: A data mining approach". *Journal of Business Research*, *69*, pp. 3341-3351.

UNIVERSITY OF WATERLOO

The inductive path components

A post on the company's Facebook page is a **unit**

The **target population** is the set of all such posts (*including all posts in the near/foreseeable future*)

The **study population** is the set of all of the posts that the company had made *to date*.

The **sample** is the set of posts in the year. There were a total of 790 such posts. Unfortunately, of these only 500 are available to us. So **our sample** is only of size 500, taken from the 790 in 2014. We don't know how these were chosen.

The company recorded 19 **variates** on each post. Only 13 of these are recorded in our dataset.

Don't really have any population attributes (yet).

The variates

For each post, values of the following 13 variates were recorded:

- `share`: the total (lifetime) number of times the post was shared
- `like`: the total (lifetime) number of times the post "liked"
- `comment`: the total (lifetime) number of comments attached to the post
- `All.interactions`: the sum of `share`, `like`, and `comment`
- `Page.likes`: the number of "likes" for the facebook page at the original time of the posting
- `Impressions`: the total (lifetime) number of times the post has been displayed, whether the post is clicked or not. The same post may be seen by a facebook user several times (e.g. via a page update in their News Feed once, whenever a friend shares it, etc.).
- `Impressions.when.page.liked`: the total (lifetime) number of times the post has been displayed to someone who has "liked" the page
- `Post.Hour`: the hour of the day at the original time of the posting (0-23)
- `Post.Weekday`: the day of the week at the original time of the posting (1-7) beginning with Sunday
- `Post.Month`: the month of the year at the original time of the posting (1-12)
- `Category`: the category of the post (as determined by two separate human reviewers according to the campaign associated with the post), one of `Action` (special offers and contests), `Product` (direct advertisement, explicit brand content), or `Inspiration` (non-explicit brand related content)
- `Type`: the type of content of the post, one of `Link`, `Photo`, `Status`, or `Video`
- `Paid`: 1 if the company paid Facebook for advertising, 0 otherwise

UNIVERSITY OF WATERLOO

The variates - explanatory or response?

It can be useful to separate variates into two different groups according to whether we are (perhaps only temporarily) thinking of them as **response** variates or as **explanatory** variates.

Here the company is curious about the effect of each posting so any variate that relates to the characteristics of the posting itself could be explanatory, whereas those that might measure the reaction of a viewer to the posting would be possible response variates.

Note that oftentimes, but not always, explanatory variates are in our control (i.e. we can either *select units* that have specific values or *assign values* to units) and response variates are not.

For example, **explanatory variates** could include those related to the time of the post (`Post.hour`, `Post.day`,`Post.month`), the `Type` of the post (one of `Link`, `Photo`, `Status`, or `Video`), the `Category` of post (one of `Action`, `Product`, or `Inspiration`), and whether the post was `Paid` advertising. Any of these could have their value assigned to the post. An explanatory variate whose value could have been somewhat selected would be the number of "likes" the Facebook page had accrued at the time of the post.

Response variates would then include the number of times the post was shared, liked, commented on, the number of times the post was displayed (`Impressions`), and the number of times to someone who indicated they had liked the page (`Impressions.when.page.liked`).

Sometimes, as with `Page.likes`, `Impressions`, `Impressions.when.page.liked`, variates might be considered as response variates for one purpose and as explantory ones for another.

The data

The first and last few rows of the data:

```
file <- path_concat(dataDirectory, "facebook.csv")
facebook <- read.csv(file)
head(facebook, n = 4)
```

```
##   All.interactions share like comment Impressions.when.page.liked Impressions
## 1              100    17   79       4                         3078        5091
## 2              164    29  130       5                        11710       19057
## 3               80    14   66       0                         2812        4373
## 4             1777   147 1572      58                        61027       87991
##   Paid Post.Hour Post.Weekday Post.Month    Category   Type Page.likes
## 1    0         3            4         12     Product  Photo     139441
## 2    0        10            3         12     Product Status     139441
## 3    0         3            3         12 Inspiration  Photo     139441
## 4    1        10            2         12     Product  Photo     139441
```

```
tail(facebook, n = 4)
```

```
##     All.interactions share like comment Impressions.when.page.liked Impressions
## 497               75    22   53       0                         3961        6229
## 498              115    18   93       4                         4742        7216
## 499              136    38   91       7                         4534        7564
## 500              119    28   91       0                         3861        7292
##     Paid Post.Hour Post.Weekday Post.Month    Category  Type Page.likes
## 497    0         8            5          1     Product Photo      81370
## 498    0         2            5          1      Action Photo      81370
## 499    0        11            4          1 Inspiration Photo      81370
## 500   NA         4            4          1     Product Photo      81370
```

UNIVERSITY OF WATERLOO

The data - summary

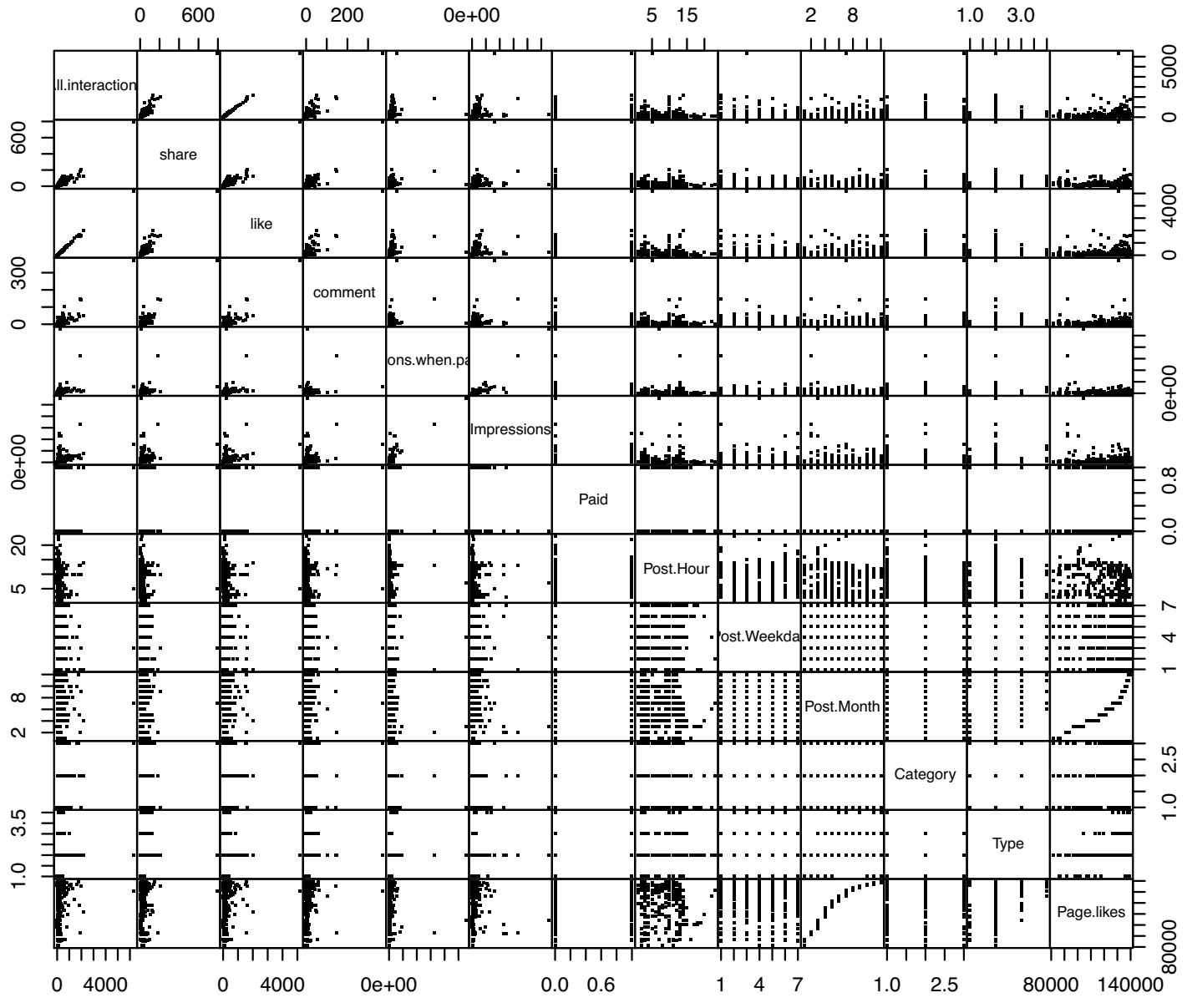
```
summary(facebook)
```

```
##  All.interactions     share              like           comment
##  Min.   :   0.0   Min.   :  0.00   Min.   :   0.0   Min.   :  0.000
##  1st Qu.:  71.0   1st Qu.: 10.00   1st Qu.:  56.5   1st Qu.:  1.000
##  Median : 123.5   Median : 19.00   Median : 101.0   Median :  3.000
##  Mean   : 212.1   Mean   : 27.27   Mean   : 177.9   Mean   :  7.482
##  3rd Qu.: 228.5   3rd Qu.: 32.25   3rd Qu.: 187.5   3rd Qu.:  7.000
##  Max.   :6334.0   Max.   :790.00   Max.   :5172.0   Max.   :372.000
##                   NA's   :4        NA's   :1
##  Impressions.when.page.liked  Impressions           Paid          Post.Hour
##  Min.   :    567             Min.   :    570   Min.   :0.0000   Min.   : 1.00
##  1st Qu.:   3970             1st Qu.:   5695   1st Qu.:0.0000   1st Qu.: 3.00
##  Median :   6256             Median :   9051   Median :0.0000   Median : 9.00
##  Mean   :  16766             Mean   :  29586   Mean   :0.2786   Mean   : 7.84
##  3rd Qu.:  14860             3rd Qu.:  22086   3rd Qu.:1.0000   3rd Qu.:11.00
##  Max.   :1107833             Max.   :1110282   Max.   :1.0000   Max.   :23.00
##                                                NA's   :1
##   Post.Weekday    Post.Month            Category       Type
##  Min.   :1.00   Min.   : 1.000   Action     :215   Link  : 22
##  1st Qu.:2.00   1st Qu.: 4.000   Inspiration:155   Photo :426
##  Median :4.00   Median : 7.000   Product    :130   Status: 45
##  Mean   :4.15   Mean   : 7.038                     Video :  7
##  3rd Qu.:6.00   3rd Qu.:10.000
##  Max.   :7.00   Max.   :12.000
##
##    Page.likes
##  Min.   : 81370
##  1st Qu.:112676
##  Median :129600
##  Mean   :123194
##  3rd Qu.:136393
##  Max.   :139441
##
```

UNIVERSITY OF WATERLOO

The data - plot summary

```
plot(facebook, gap = 0, pch =".")
```



UNIVERSITY OF WATERLOO

The data - interactive plot summaries

```
library(loon)
findCatVars <- function(data) {
    isCatVar <- sapply(names(data),
                       FUN  = function(name) {is.factor(data[,name])})
    catVars <- names(data)[isCatVar]
    catVars
}

l_barPlots <- function(data, linkingGroup,  together = TRUE){
    if(missing(linkingGroup)) {
        linkingGroup <- deparse(substitute(data)) # use the data frame name
        }
    catVars <- findCatVars(data)
    if (together){
        parent <- tktoplevel()
        tktitle(parent) <- "Counts for factors"
        nrows <- floor(sqrt(length(catVars)))
        ncols <- ceiling(sqrt(length(catVars)))
        row <- 0
        col <- 0
        for (var in catVars) {
            barplot <- l_hist(data[,var],
                              linkingGroup = linkingGroup,
                              title = var,
                              xlabel = var,
                              parent = parent)
            if (col >= ncols){
                row <- row + 1
                col <- 0}
            tkgrid(barplot, row = row, column = col, sticky = "nesw")
            col <- col + 1}
        # Continued next slide
        # ...
```

The data - interactive plot summaries

```
        # ...
        # Continued from previous slide
        # Configure columns to resize with window
        for (col in 0:(ncols-1)){
            tkgrid.columnconfigure(parent, col, weight = 1)}
        # Configure rows to resize with window
        for (row in 0:(nrows-1)){
            tkgrid.rowconfigure(parent, row, weight = 1)}
    } else {
        for (var in catVars) {
            barplot <- l_hist(data[,var],
                              linkingGroup = linkingGroup,
                              title = var,
                              xlabel = var)

        }
    }
}
```

Can now call `l_barPlots()` on our data set:

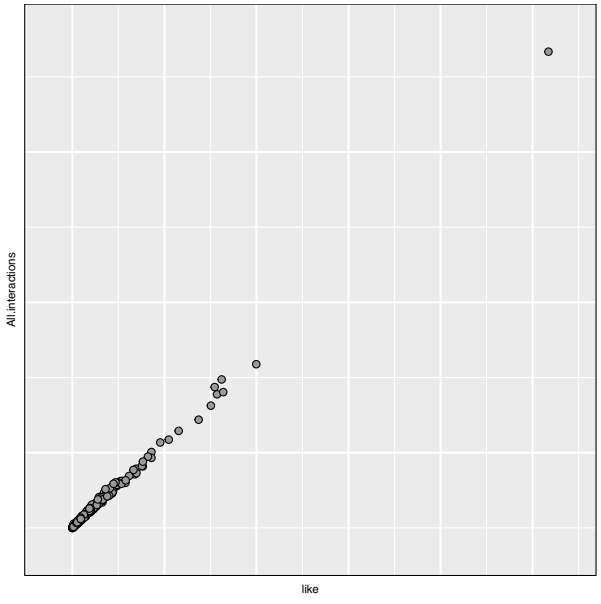
```
bp <- l_barPlots(facebook)
```

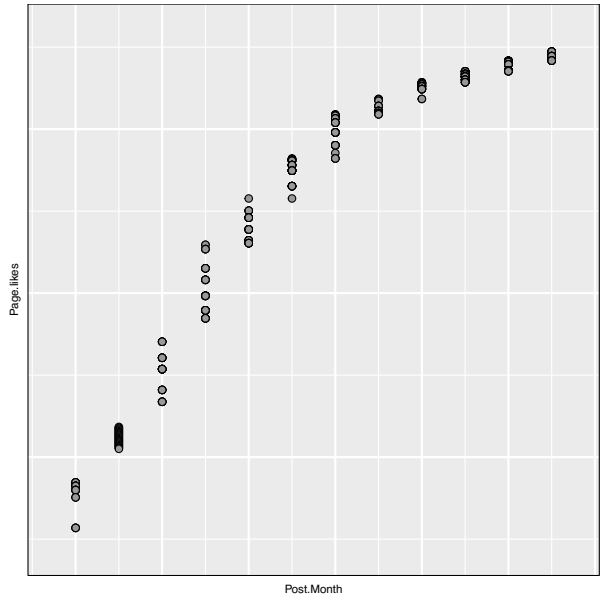
And a pairs plot

```
vars <- setdiff(names(facebook), findCatVars(facebook))
pp <- l_pairs(facebook[, vars], glyph = "ocircle", size = 2,
              showHistograms = TRUE,
              linkingGroup = "facebook")
```

Relationships between pairs of variates

From the plots there are some obvious fairly strong relationships. Notably,





But then, `All.interactions` = sum of `share`, `like`, and `comment` which appears to be dominated by `like`.

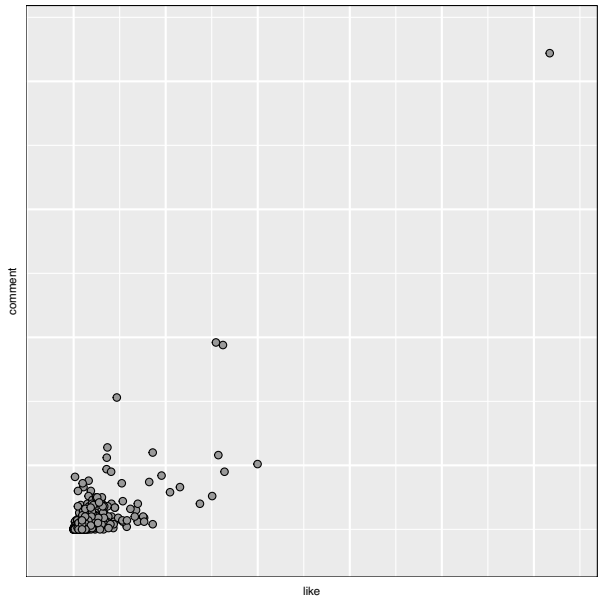
Also, `Page.likes` are total Facebook page likes at the time of posting. The number of `Page.likes` is necessarily non-decreasing in time.

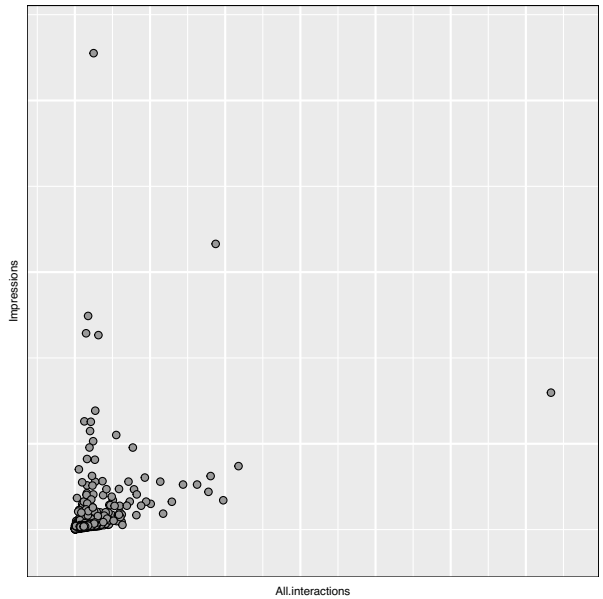
In both cases, it would seem that there is a very strong relation between the variates in each pair. One or the other of each pair might be discarded.

UNIVERSITY OF WATERLOO

Relationships between pairs of variates

For many other plots there also appears to be a relationship, though fairly weak. For example,





Transforming variates

All of the response variates appear to have severely skewed distributions (e.g. `share`, `like`, etc.). We might consider a monotonic (and order preserving) transformation such as the logarithm.

Note that zero occurs in the counts for `All.interactions`, `share`, `like`, and `comment`, so we add a small amount first (+1) to all of these.

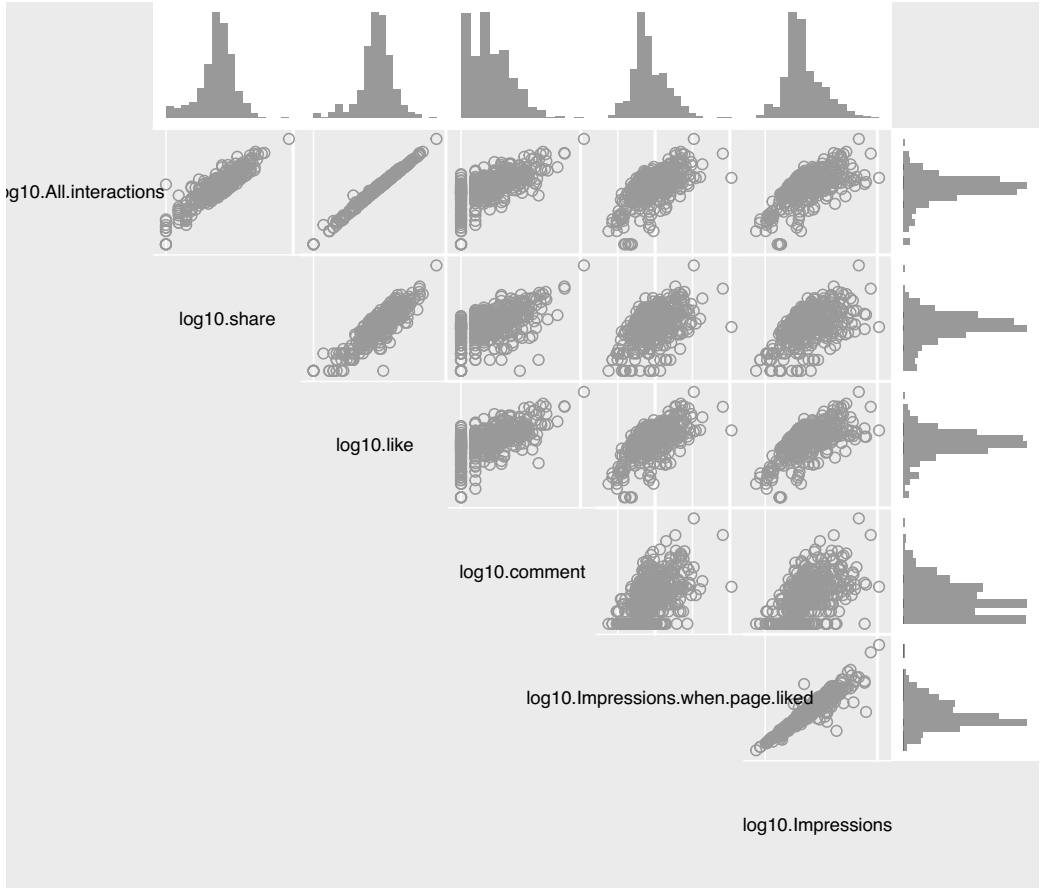
```
facebooklog10 <- data.frame(
       log10(1 +
             facebook[, c("All.interactions", "share", "like", "comment")]),
       log10(facebook[, c("Impressions.when.page.liked", "Impressions")])
)
names(facebooklog10) <- paste("log10", names(facebooklog10), sep = ".")
head(facebooklog10)
```

```
##   log10.All.interactions log10.share log10.like log10.comment
## 1               2.004321    1.255273   1.903090     0.6989700
## 2               2.217484    1.477121   2.117271     0.7781513
## 3               1.908485    1.176091   1.826075     0.0000000
## 4               3.249932    2.170262   3.196729     1.7708520
## 5               2.595496    1.698970   2.513218     1.3010300
## 6               2.271842    1.531479   2.184691     0.3010300
##   log10.Impressions.when.page.liked log10.Impressions
## 1                          3.488269          3.706803
## 2                          4.068557          4.280055
## 3                          3.449015          3.640779
## 4                          4.785522          4.944438
## 5                          3.794349          4.133347
## 6                          4.205042          4.319085
```

UNIVERSITY OF WATERLOO

Transforming variates

```
pplog10 <- l_pairs(facebooklog10, glyph = "ocircle",
                   linkingGroup = "facebook", showHistograms = TRUE)
plot(pplog10)
```

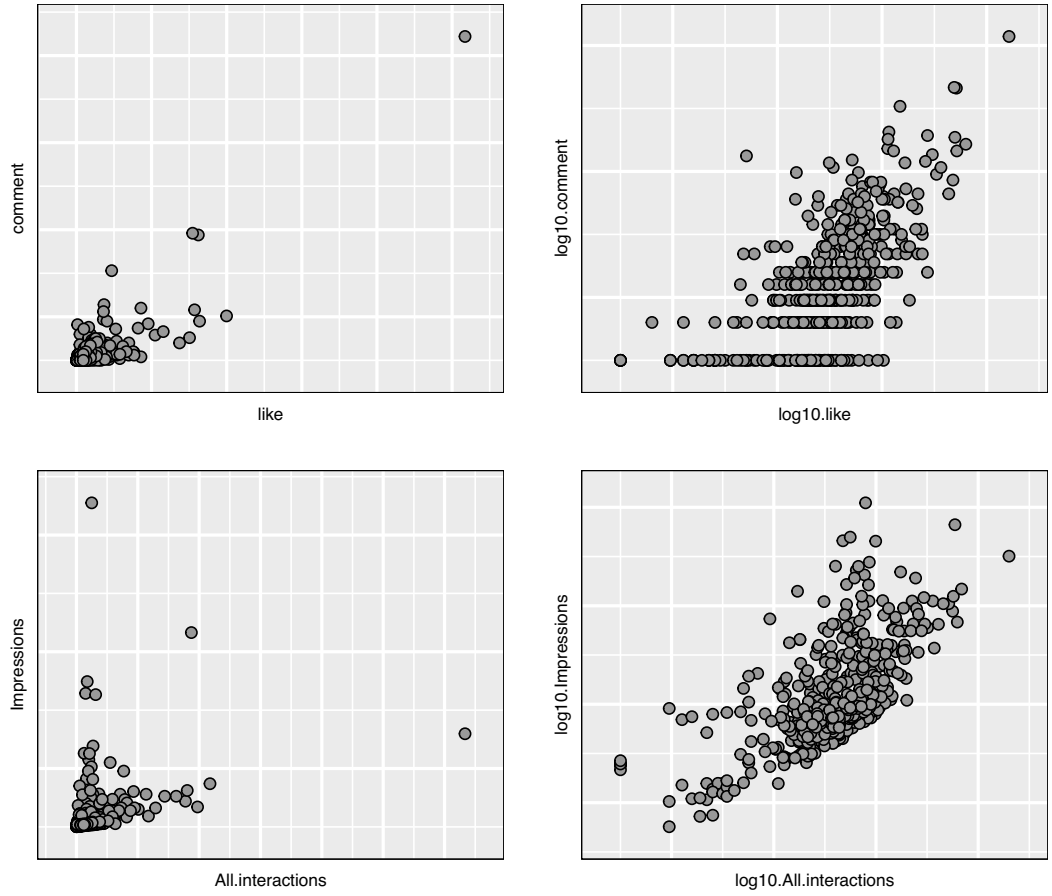


Some histograms look more symmetric, less skewed. Others look skewed in the opposite direction.

Similarly, some plots look more like straight lines; others look like they are now spread out a lot on the left instead of the right.

Transforming variates

For example, compare these plots:



Is there some way of not going too far?

The family of power transformations

When $x > 0$, one possible "family" of transformations is the power family.

For $x > 0$, we can write this family (over the power α) as

$$T_\alpha(x) = \begin{cases} ax^\alpha + b & (\alpha \neq 0) \\ c\log(x) + d & (\alpha = 0) \end{cases}$$

where a, b, c, d and α are real numbers with $a > 0$ when $\alpha > 0$, $a < 0$ when $\alpha < 0$, and $c > 0$ when $\alpha = 0$. The choices of a, b, c and d are somewhat arbitrary otherwise.

Since location and scale are not important in determining the **shape** of the distribution, we might settle on particular choices which are more mathematicallly convenient. For example,

$$T_\alpha(x) = \frac{x^\alpha - 1}{\alpha} \qquad \forall\ \alpha$$

which requires no separate equation for the $x = 0$ case, since $lim_{\alpha \to 0} T_\alpha(x) = log(x)$.

The family of power transformations

We could have different values of α, say α_x and α_y, respectively for x and y points in a scatterplot. In loon , we can create an interactive tool that allows us to view a scatterplot of pairs $(T_{\alpha_x}(x), T_{\alpha_y}(y))$ as we vary α_x and α_y.

The code looks like this:

```
power_xy <- function(x, y=NULL, xlab=NULL, ylab=NULL,
                     linkingGroup,
                     linkingKey, from = -5, to = 5,
                     ...) {
    if (is.null(xlab)) {xlab <- "x"}
    if (is.null(ylab)) {ylab <- "y"}
    if (missing(linkingKey)) {linkingKey <- paste0(seq(0, length(x) -1))}
    if (missing(linkingGroup)) {linkingGroup <- deparse(substitute(x))}

    powerfun <- function(x, alpha) {
        if (alpha == 0) log(x) else (x^alpha-1)/alpha}

    scale01 <- function(x) {
        minx <- min(x, na.rm = TRUE)
        maxx <- max(x, na.rm = TRUE)
        (x-minx)/(maxx - minx) + 1}

    tt <- tktoplevel()
    xs <- scale01(x)
    ys <- scale01(y)
    # CONTINUED ON NEXT SLIDE
```

The family of power transformations

```
# CONTINUED FROM PREVIOUS
# scatterplot
p <- l_plot(x=xs, y=ys, xlabel=xlab, ylabel=ylab,
            linkingGroup = linkingGroup,
            linkingKey = linkingKey,
            parent=tt, ...)
# Save the x and y values from p
xy_p <- data.frame(x = p["x"], y = p["y"])

fit <- lm(y ~ x, data = xy_p)
# layer fit
xrng <- range(xy_p$x)
yhat <- predict(fit, data.frame(x = xrng))
line <- l_layer_line(p, x = xrng, y = yhat, linewidth = 3, index = "end")
# histogram on each
h_x <- l_hist(xs, xlabel=xlab, yshows="density",
              linkingGroup = linkingGroup, linkingKey = linkingKey)
h_y <- l_hist(ys, xlabel=ylab, yshows="density",
              linkingGroup = linkingGroup, linkingKey = linkingKey,
              swapAxes = TRUE)
# save the original histogram values
h_x_vals <- h_x["x"]
h_y_vals <- h_y["x"]
# CONTINUED ON NEXT SLIDE
```

The family of power transformations

```
# CONTINUED FROM PREVIOUS SLIDE

# Set up power transformations
alpha_x <- tclVar('1')
alpha_y <- tclVar('1')

# Create two slider scales for the two powers
# Reverse to and from because scales are vertical
sx <- tkscale(tt, orient='vertical', label='power x', variable=alpha_x,
              from=to, to=from, resolution=0.1)
sy <- tkscale(tt, orient='vertical', label='power y', variable=alpha_y,
              from=to, to=from, resolution=0.1)

# pack the sliders into the same window as the scatterplot
tkpack(sy, sx, fill='y', side='right')
tkpack(p, fill='both', expand=TRUE)

# CONTINUED on NEXT SLIDE
```

The family of power transformations

```
    # CONTINUED FROM PREVIOUS SLIDE
    # the update function, called when sliders move
    update <- function(...) {
        ## powers
        alphax <- as.numeric(tclvalue(alpha_x))
        alphay <- as.numeric(tclvalue(alpha_y))

        ## labels
        xlabel <- if (alphax==0) {
            paste0("log(",xlab,")")
        } else {
            if (alphax==1) {
                xlab
            } else {
                paste0(xlab,"^",alphax)
            }
        }
        ylabel <- if (alphay==0) {
            paste0("log(",ylab,")")
        } else {
            if (alphay==1) {
                ylab
            } else {
                paste0(ylab,"^",alphay)
            }
        }

        # CONTINUED on NEXT SLIDE
```

The family of power transformations

```
# CONTINUED FROM PREVIOUS SLIDE
#
# new plot x and y
p_xnew <- scale01(powerfun(xy_p$x, alphax))
p_xnew_range <- range(p_xnew)
p_ynew <- scale01(powerfun(xy_p$y, alphay))

# update the fitted line values
fit.temp <- lm(p_ynew ~ p_xnew)
yhat <- predict(fit.temp, data.frame(p_xnew = p_xnew_range))
l_configure(line, y = yhat, x = p_xnew_range)

# update plot
l_configure(p,
            x = p_xnew,
            y = p_ynew,
            xlabel = xlabel,
            ylabel = ylabel
)
l_scaleto_world(p)

# CONTINUED on NEXT SLIDE
```

The family of power transformations

```
        # CONTINUED FROM PREVIOUS SLIDE
        # update the histograms
        binwidthx <- h_x['binwidth']
        binwidthy <- h_y['binwidth']
        h_x_vals_new <- scale01(powerfun(h_x_vals, alphax))
        h_y_vals_new <- scale01(powerfun(h_y_vals, alphay))
        l_configure(h_x, x = h_x_vals_new,
                    binwidth = binwidthx,
                    #origin = originx,
                    yshows="density",
                    xlabel= xlabel)
        l_configure(h_y, x = h_y_vals_new,
                    binwidth = binwidthy,
                    #origin = originy,
                    yshows="density",
                    xlabel= ylabel)
        l_scaleto_plot(h_x)
        l_scaleto_plot(h_y)
    }
    # end of update function
    # attach the update to the sliders
    tkconfigure(sx, command=update)
    tkconfigure(sy, command=update)
    invisible(p)
}
```

UNIVERSITY OF WATERLOO

The family of power transformations

We can now try this function out in loon .

```
# Need to get a version of facebook and if necessary remove nas with na.omit()
#
# First add 1 where necessary
#
facebook1 <- facebook
facebook1[,c("All.interactions", "share",
             "like", "comment")] <- 1 + facebook[, c("All.interactions", "share",
                                                    "like", "comment")]

#
# facebook1 <- na.omit(facebook1) # USE THIS
#

p_MonthPageLikes <- with(facebook1,
                         power_xy(x=Post.Month, y=Page.likes,
                                  xlab="month of post", ylab="number page likes",
                                  title = "Page likes vs month",
                                  linkingGroup = "facebook",
                                  itemLabel = paste0("Category: ", Category, "\n",
                                                     "Type: ", Type, "\n",
                                                     "Paid: ",
                                                     c("no", "yes")[Paid]),
                                  showItemLabels = TRUE))
```

The family of power transformations

Try this function out on the "like" and "share" variates. We need a wider range of powers.

```
p_likeShare <- with(facebook1,
                    power_xy(x=like, y=share, xlab="likes", ylab="shares",
                             from = -30, to = 10,
                             title = "shares vs likes",
                             linkingGroup = "facebook",
                             itemLabel = paste0("Category: ", Category, "\n",
                                                "Type: ", Type, "\n",
                                                "Paid: ", c("no", "yes")[Paid]),
                             showItemLabels = TRUE))
```

Tukey's ladder of power transformations

```
#  A convenient mnemonic  ....
#
#  Tukey's "ladder" of power transformations
# ... -2, -1, -1/2, -1/3, 0, 1/3, 1/2, 1, 2, ...
#   <--- down             up -->        ^start
#
#     Or, as a ladder:
#
#                    Power
#                  ---------
#                      .
#                      .
#                      .
#                  ---------
#                      2
#                  ---------
#                      1       <-- raw data (start)
#                  ---------
#                     1/2
#                  ---------
#                      0       <-- logarithm
#                  ---------
#                    -1/2
#                  ---------
#                     -1
#                  ---------
#                     -2
#                  ---------
#
#                     Etc.
```

John Tukey suggested imagining that the powers were arranged in a "ladder" with the smallest powers on the bottom and the largest on the top.

UNIVERSITY OF WATERLOO

Tukey's ladder of power transformations - Bump rules

Bump rule 1 for making densities (histograms) more symmetric.

The location of the density's bump (mode) tells you which way to "move" on the ladder.

- if it is concentrated on "lower" values, then move the power "lower" on the ladder
  - i.e. move down the ladder
- if it is concentrated on "higher" values, then move the power "higher" on the ladder
  - i.e. move up the ladder

Bump rule 2 for straightening scatterplots.

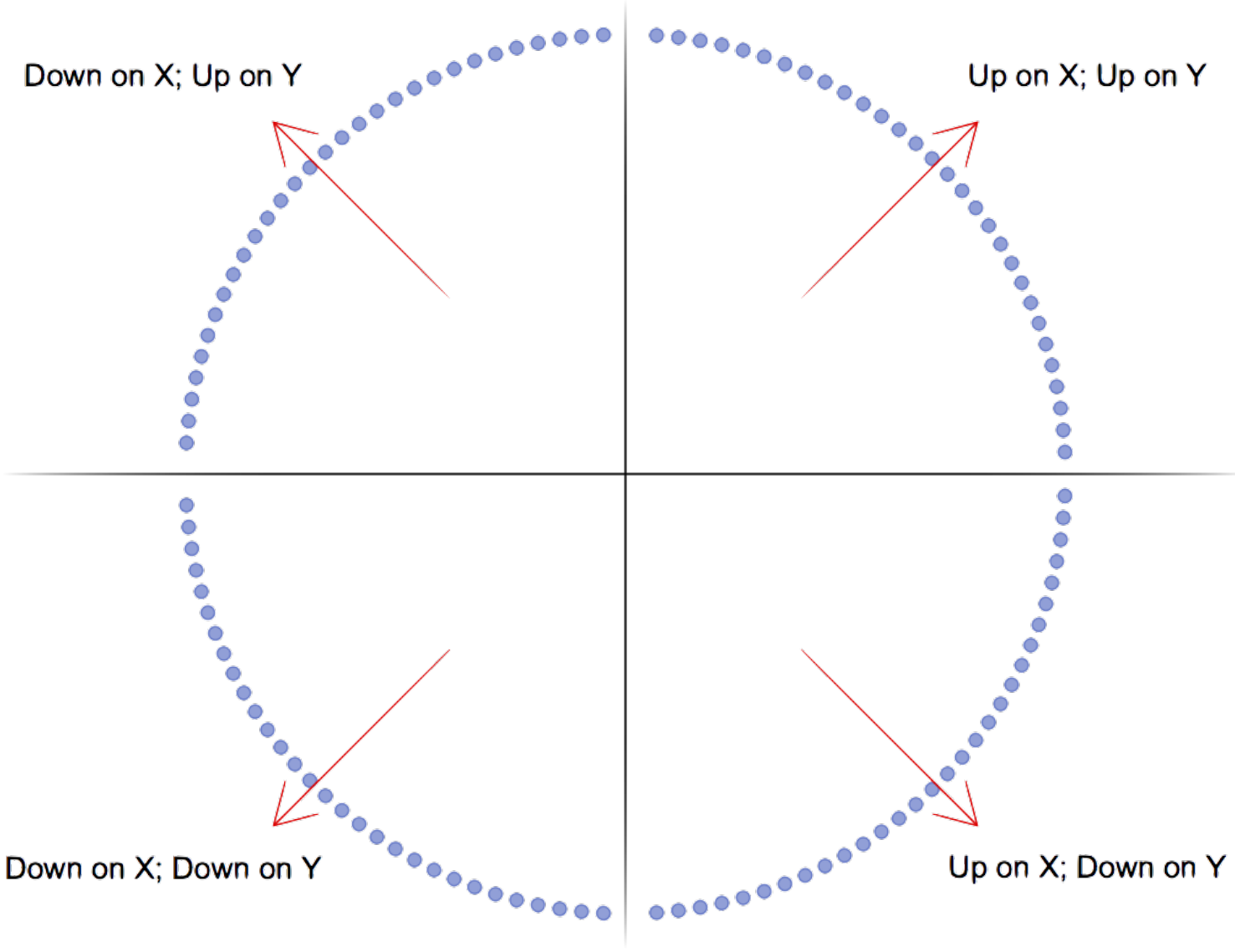
Bump is now the direction of the curve of the scatterplot,

Bump points to a direction on the x axis AND on a (possibly different) direction on the y axis.

- Bump pointing up (or down) on the x axis means move up (or down) on α_x ladder.
- Bump pointing up (or down) on the y axis means move up (or down) on α_y ladder.

Tukey's ladder of power transformations - Bump rules

Monotonic curved relations will look like one of the four quadrants



UNIVERSITY OF WATERLOO

Fitting models to data

Consider, for example, the plots and fitted lines produced by the code below.

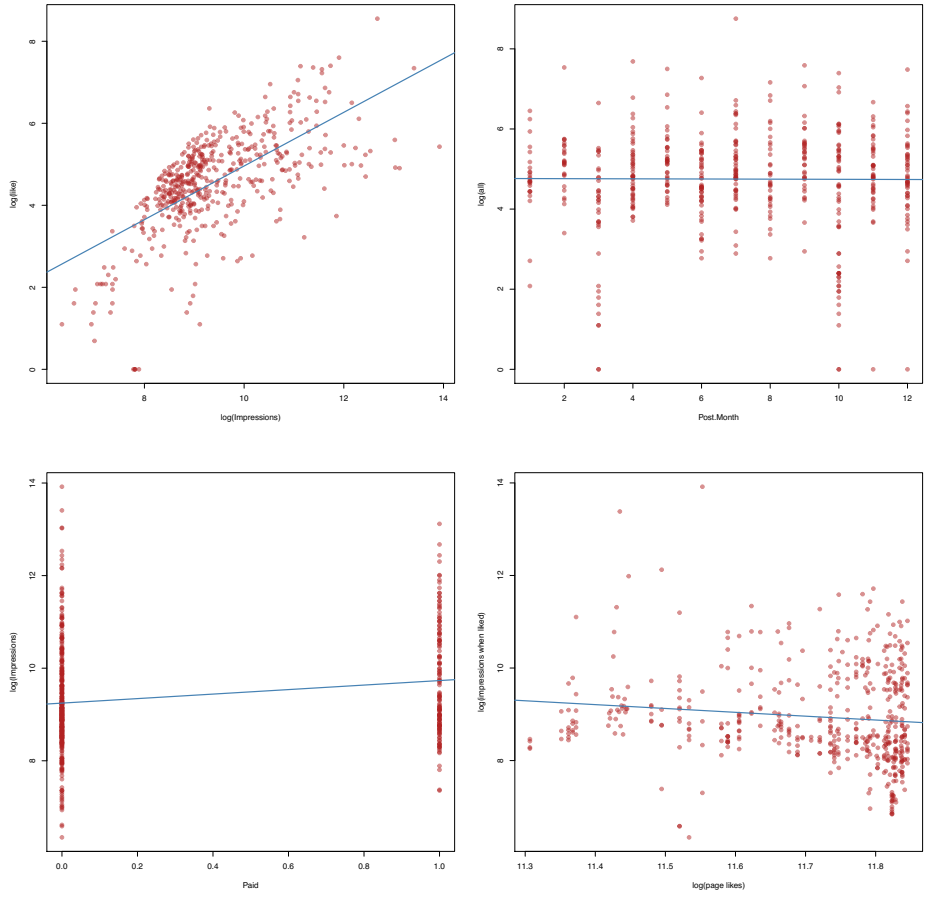
```
fit1 <- lm(log(like + 1) ~ log(Impressions), data = facebook)
fit2 <- lm(log(All.interactions +1) ~ Post.Month, data = facebook)
fit3 <- lm(log(Impressions) ~ Paid, data = facebook)
fit4 <- lm(log(Impressions.when.page.liked) ~ log(Page.likes + 1), data = facebook)

savePar <- par(mfrow=c(2,2))
colour <- adjustcolor("firebrick", 0.5)
with(facebook,
     { plot(log(Impressions), log(like + 1), ylab = "log(like)", pch=19, col=colour)
       abline(fit1, col="steelblue", lwd=2)
       plot(Post.Month, log(All.interactions +1), ylab = "log(all)", pch=19, col=colour)
       abline(fit2, col="steelblue", lwd=2)
       plot(Paid, log(Impressions), pch=19, col=colour)
       abline(fit3, col="steelblue", lwd=2)
       plot(log(Page.likes + 1), log(Impressions.when.page.liked),
            ylab = "log(impressions when liked)", xlab = "log(page likes)",
            pch=19, col=colour)
       abline(fit4, col="steelblue", lwd=2)
       }
    )
par(savePar)
```

UNIVERSITY OF WATERLOO

Fitting models to data

Results are:



UNIVERSITY OF WATERLOO

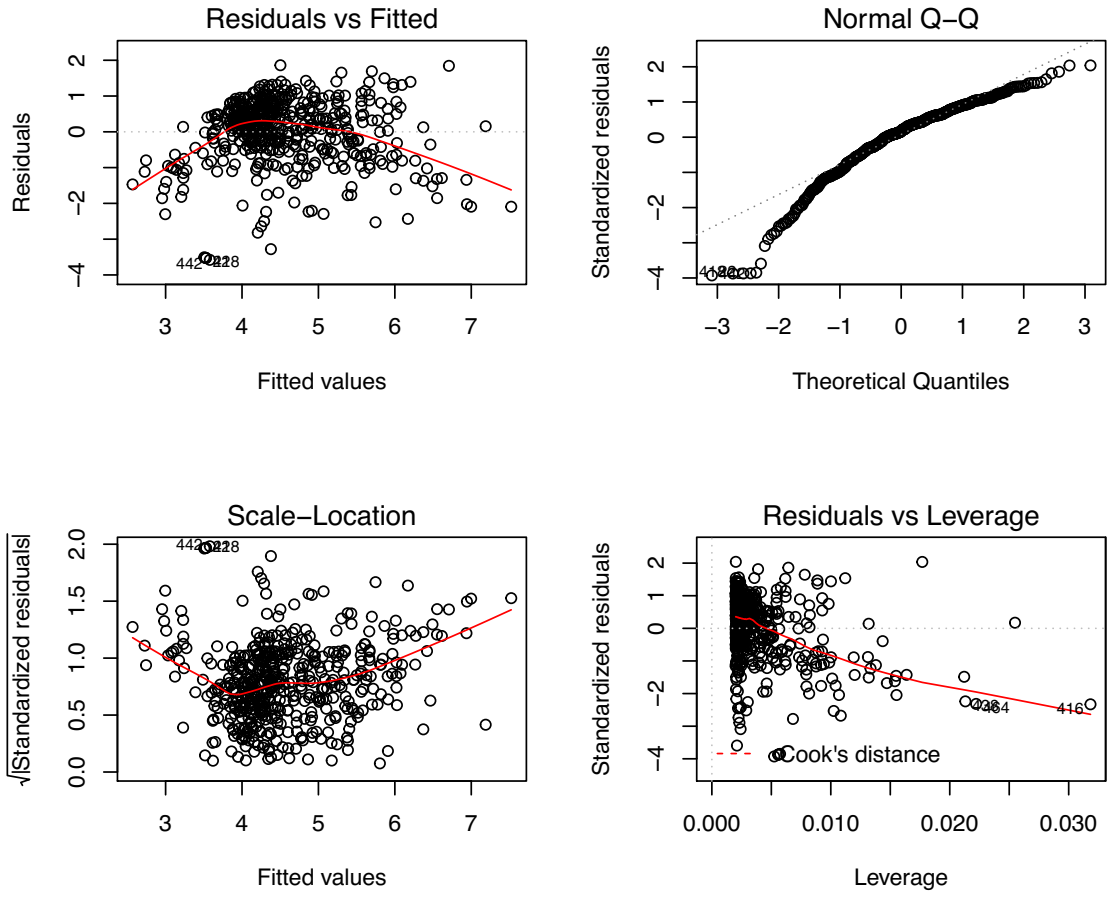
First fit results:

```
summary(fit1)
```

```
##
## Call:
## lm(formula = log(like + 1) ~ log(Impressions), data = facebook)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5784 -0.4542  0.1555  0.5969  1.8605
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.58284    0.32893  -4.812 1.98e-06 ***
## log(Impressions)  0.65416    0.03478  18.808  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9135 on 497 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.4158, Adjusted R-squared:  0.4146
## F-statistic: 353.7 on 1 and 497 DF,  p-value: < 2.2e-16
```

First fit results:

```
oldPar <- par(mfrow=c(2,2))
plot(fit1)
```




```
par(oldPar)
```

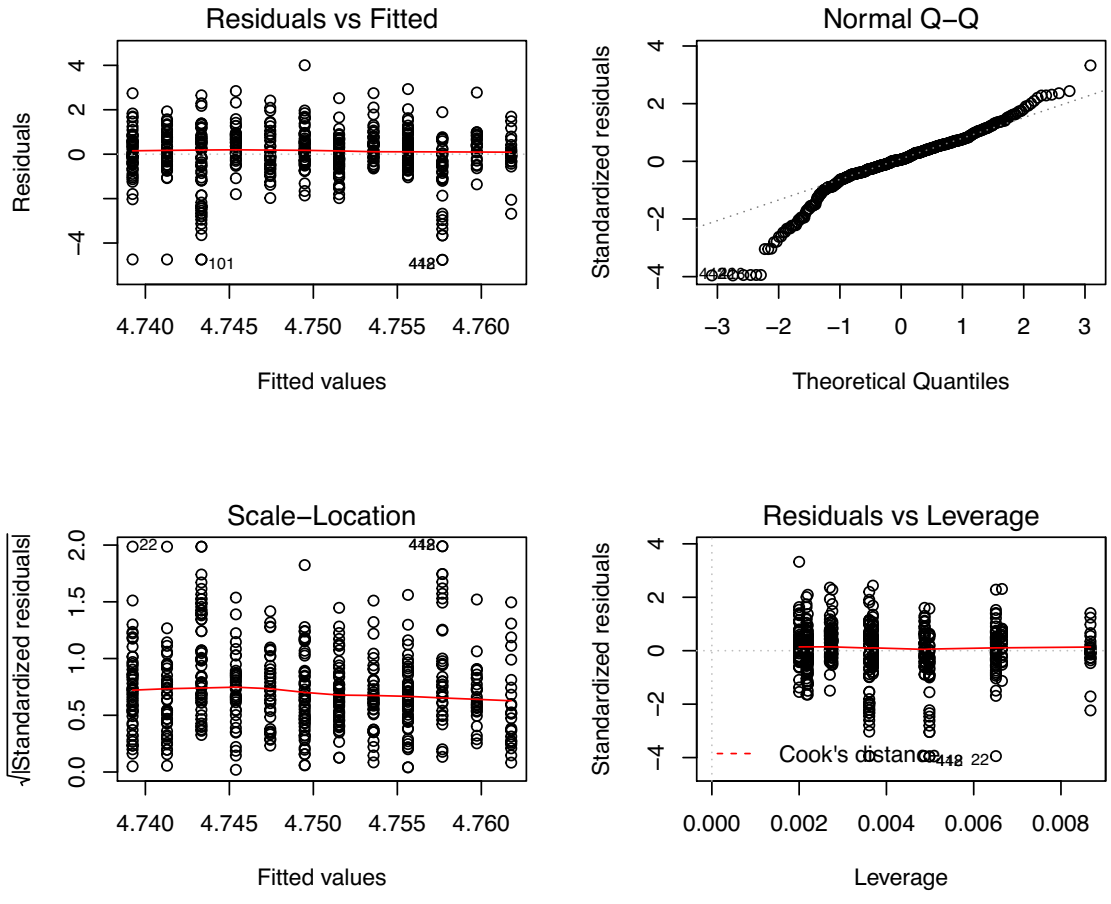
Second fit results:

```
summary(fit2)
```

```
##
## Call:
## lm(formula = log(All.interactions + 1) ~ Post.Month, data = facebook)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.7577 -0.4743  0.0746  0.6856  4.0044
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.763830   0.126806  37.568   <2e-16 ***
## Post.Month  -0.002049   0.016309  -0.126      0.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.205 on 498 degrees of freedom
## Multiple R-squared:  3.17e-05,   Adjusted R-squared:  -0.001976
## F-statistic: 0.01579 on 1 and 498 DF,  p-value: 0.9001
```

Second fit results:

```
oldPar <- par(mfrow=c(2,2))
plot(fit2)
```




```
par(oldPar)
```

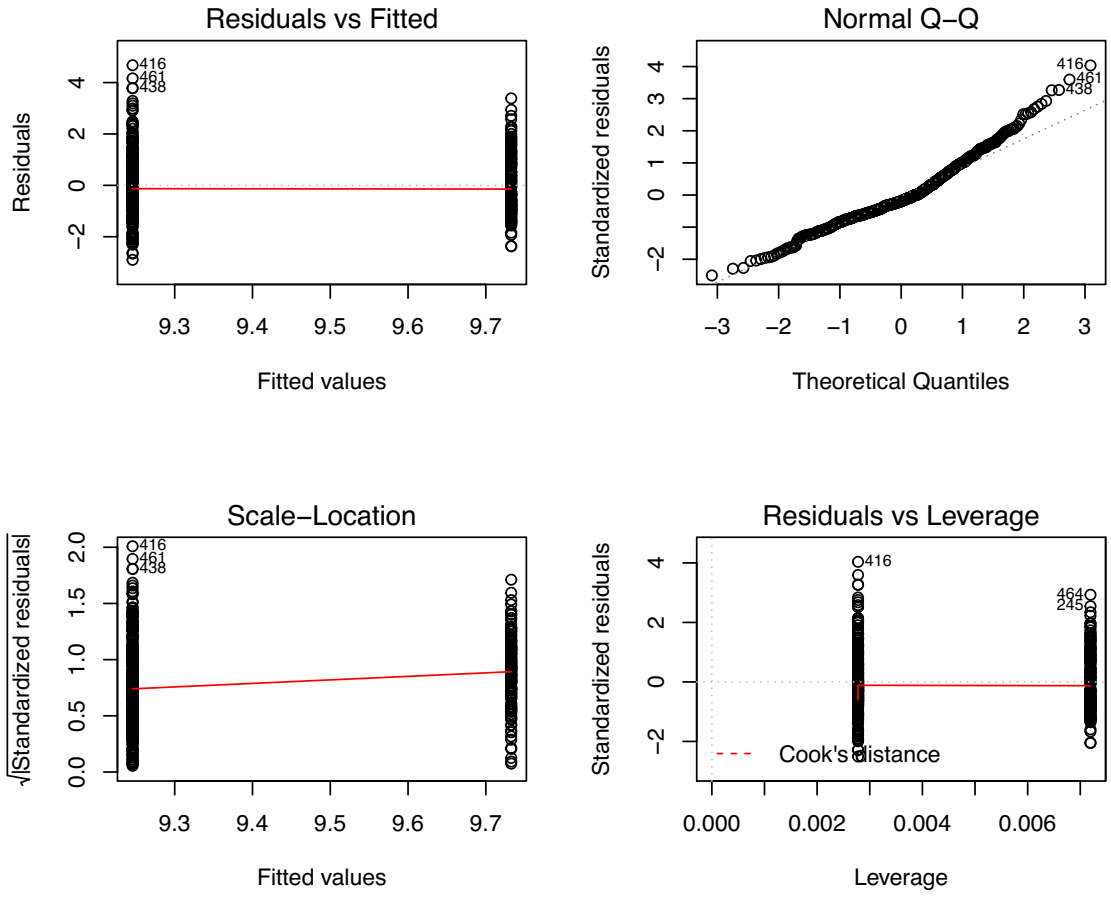
Third fit results:

```
summary(fit3)
```

```
##
## Call:
## lm(formula = log(Impressions) ~ Paid, data = facebook)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.9002 -0.7373 -0.2215  0.6549  4.6743
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9.24586    0.06114 151.236  < 2e-16 ***
## Paid         0.48681    0.11583   4.203 3.13e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.16 on 497 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.03432,    Adjusted R-squared:  0.03238
## F-statistic: 17.66 on 1 and 497 DF,  p-value: 3.128e-05
```

Third fit results:

```
oldPar <- par(mfrow=c(2,2))
plot(fit3)
```




```
par(oldPar)
```

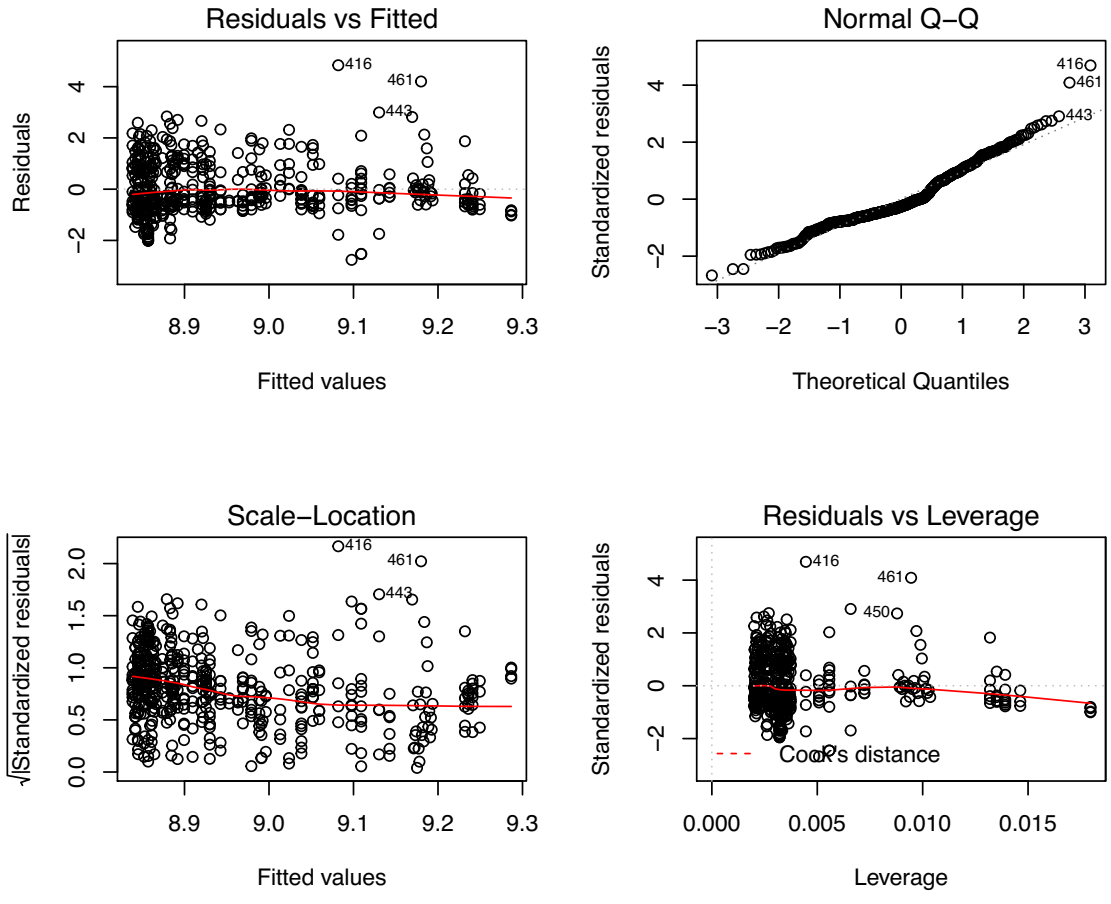
Fourth fit results:

```
summary(fit4)
```

```
##
## Call:
## lm(formula = log(Impressions.when.page.liked) ~ log(Page.likes +
##     1), data = facebook)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7574 -0.6522 -0.2636  0.6733  4.8359
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           18.6926     3.7743   4.953    1e-06 ***
## log(Page.likes + 1)  -0.8319     0.3222  -2.582   0.0101 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.033 on 498 degrees of freedom
## Multiple R-squared:  0.01321,    Adjusted R-squared:  0.01122
## F-statistic: 6.664 on 1 and 498 DF,  p-value: 0.01012
```

Fourth fit results:

```
oldPar <- par(mfrow=c(2,2))
plot(fit4)
```




```
par(oldPar)
```

How about:

```
fit5 <- lm(I(Page.likes^3) ~ sqrt(Post.Month), data = facebook)
```

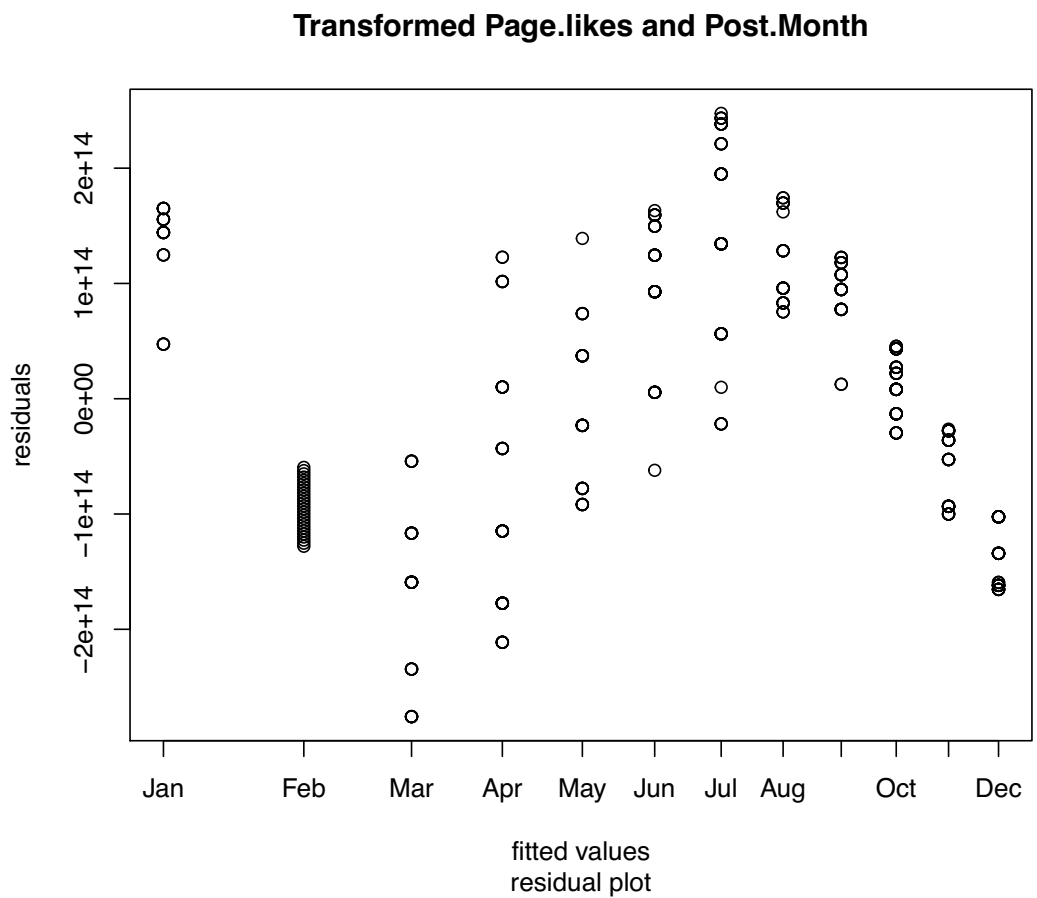
```
summary(fit5)
```

```
##
## Call:
## lm(formula = I(Page.likes^3) ~ sqrt(Post.Month), data = facebook)
##
## Residuals:
##        Min          1Q      Median          3Q         Max
## -2.757e+14 -1.000e+14  5.535e+12  9.590e+13  2.475e+14
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -4.510e+14  2.039e+13  -22.12   <2e-16 ***
## sqrt(Post.Month)   9.424e+14  7.687e+12  122.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.186e+14 on 498 degrees of freedom
## Multiple R-squared:  0.9679, Adjusted R-squared:  0.9679
## F-statistic: 1.503e+04 on 1 and 498 DF,  p-value: < 2.2e-16
```

Fitting models to data

Fifth fit results:

```
plot(fit5$fitted.values, fit5$residuals,
     xlab = "fitted values", ylab = "residuals",
     main = "Transformed Page.likes and Post.Month",
     sub = "residual plot", xaxt ="n"
)
axis(side=1,  labels = month.abb,
     at = predict(fit5,
                  newdata =
                      data.frame(Post.Month = sort(unique(facebook$Post.Month)))))
```



UNIVERSITY OF WATERLOO